Strongly polynomial algorithm for the intersection of a line with a polymatroid

Jean Fonlupt, Alexandre Skoda

Strongly polynomial algorithm for the intersection of a line with a polymatroid

Jean Fonlupt[1] and Alexandre Skoda[2]

[1] Equipe Combinatoire et Optimisation. CNRS et Université Paris 6
jean.fonlupt@math.jussieu.fr
[2] Equipe Combinatoire et Optimisation. CNRS et Université Paris 6
France Telecom R&D. Sophia Antipolis**
askoda@math.jussieu.fr

Summary. We present a new algorithm for the problem of determining the intersection of a half-line $\Delta_u = \{x \in \mathbb{R}^N \mid x = \lambda u \text{ for } \lambda \geq 0\}$ with a polymatroid. We then propose a second algorithm which generalizes the first algorithm and solves a parametric linear program. We prove that these two algorithms are strongly polynomial and that their running time is $O(n^8 + \gamma n^7)$ where γ is the time for an oracle call. The second algorithm gives a polynomial algorithm to solve the submodular function minimization problem and to simultaneously compute the strength of a network with complexity bound $O(n^8 + \gamma n^7)$.

Key words: Algorithm, graph, strength of a graph, submodular function, matroid, polymatroid, parametric linear programming.

1 Introduction

Let f be a *set function* on a finite set $N = \{1, \ldots, n\}$. f is *submodular* if

$$f(S \cap T) + f(S \cup T) \leq f(S) + f(T)$$

for all subsets S, T of N. If f is also *normalized* ($f(\emptyset) = 0$) and *monotone* ($f(U) \leq f(T)$ whenever $U \subseteq T$), the *polymatroid associated with* f is defined by:

$$P(f) = \{x \in \mathbb{R}^N \mid x \geq 0,\ x(T) \leq f(T) \text{ for each } T \subseteq N\}.$$

** This research has been supported by a grant of "France Telecom R&D Sophia Antipolis" during three years.

Let u be a vector of $\mathbb{R}^N$ which describes the direction of the half-line $\Delta_u = \{x \in \mathbb{R}^N \mid x = \lambda u \text{ for } \lambda \geq 0\}$. $P(f) \cap \Delta_u$ is a closed interval $[0, \lambda_{max}]u$ where λ_{max} is the value of the linear program:

$$\max \{\lambda \mid x \in P(f) \cap \Delta_u\} \qquad (\mathcal{P})$$

This problem will be called the *Intersection Problem*.
We propose a strongly polynomial combinatorial algorithm for solving $\mathcal{P}$. This algorithm called the *Intersection Algorithm* runs in time $O(n^8 + \gamma n^7)$ where γ is the time of an oracle call.
The Intersection Problem has interesting applications: consider the following parametric linear program:

$$z(\lambda) = \max \left\{ \sum_{i \in N} x_i \mid x \in \mathrm{P(f)},\ x \leq \lambda u \right\} \qquad (\mathcal{P}_\lambda)$$

Standard results in parametric linear programming show that the function $\lambda \longrightarrow z(\lambda)$ is non-decreasing, concave and piecewise linear.
We prove that we need to solve at most n times the Intersection Problem to obtain a complete description of $z(\lambda)$. This will be the object of the *Parametric Intersection Algorithm* which is a simple variant of the Intersection Algorithm. This second algorithm runs also in time $O(n^8 + \gamma n^7)$. As the solution of the parametric linear program for a certain choice of the parameter λ solves the *submodular function minimization* problem and for another choice of this parameter provides the value of the *Strength of a Network* if f is the rank function of a graphic matroid of a graph, we obtain a strongly polynomial algorithm with complexity bound $O(n^8 + \gamma n^7)$ for solving simultaneously these two problems.
Polymatroids were introduced by Edmonds [6]. Examples of submodular functions are described in Fleischer [7], Fujishige [10], Korte and Vygen [14], Schrijver [20]. The Intersection Problem can be solved in strongly polynomial time using the ellipsoid method [11]. Schrijver [19] and Iwata, Fleischer and Fujishige [13] independently proposed combinatorial strongly polynomial algorithms for submodular function minimization. Vygen [22] showed that the running time of Schrijver's original algorithm was $O(n^8 + \gamma n^7)$ which was also the improvement given by Fleischer and Iwata [8]. Iwata [12] improved this bound to $O((n^7 + \gamma n^6) \log n)$. Orlin [18] developed a strongly polynomial algorithm in $O(n^6 + \gamma n^5)$ which is the best complexity bound known so far.
All these approaches depend on the decomposition of feasible points of the polymatroid as convex combination of extreme points of this polytope; this idea was originated by Cunningham in [3] and [5]. Our own approach is based on the polymatroid itself and not on the base polyhedron associated to the

polymatroid as do the above-cited authors; this is a major difference. Note however that an n-dimensional polymatroid can be seen as a projection of an $(n+1)$-dimensional base polyhedron (see Fujishige [10]) but this observation can not be easily exploited from an algorithmic point of view.
As a generalization of lexicographically optimal flow of Megiddo [15], Fujishige [9] introduced the concept of lexicographically optimal base of a polymatroid. Problem $\mathcal{P}_\lambda$ is known to be equivalent to the lexicographically optimal base problem [10] but we will not use this result.
Cunningham [4] in his study of the strength of a network had an approach based on the forest polytope of a graph. In [2], Cheng and Cunningham studied, for the problem of the strength of a network, a parametric linear program which can be seen as a specific case of our parametric linear program.
Nagano [16] developped a strongly polynomial algorithm for the intersection of an extended polymatroid and a straight line, based on a direct application of the submodular function minimization problem. Although our algorithm is only designed for the intersection with a polymatroid, it provides a new algorithm for the submodular function minimization problem. Recently Nagano [17], extending Orlin's algorithm [18] proposed an algorithm in $O(n^6 + \gamma n^5)$ to solve the parametric problem $\mathcal{P}_\lambda$. Our algorithm is not as good as Nagano's in time complexity but we hope that the interest of our approach is justified by the method we employ : we work directly on the polymatroid itself and we do not use known submodular function minimization algorithms.
Many of the results presented here appear in the Ph.D thesis of Skoda [21].
The content of the paper can be summarized as follows. Section 2 recalls basic definitions and properties of polymatroids and present the general framework of the Intersection Algorithm. Section 3 describes the Intersection Algorithm and evaluates the time complexity of this algorithm. Section 4 describes the Parametric Intersection Algorithm and proves the strongly polynomiality of this algorithm; it is also shown in section 4 why this algorithm simultaneously solve the submodular minimization problem and finds the strength of a network. Many of the results in section 4 are standard (see for instance Fujishige [10]) but we include them with short proofs for completeness. Section 5 summarizes the results of the previous sections and proposes some new problems.

2 Preliminaries

1. If $\emptyset \subseteq T \subseteq N$, define $f_{N\setminus T}$ by

$$f_{N\setminus T}(A) = f(A) \quad \text{for all } \emptyset \subseteq A \subseteq N \setminus T.$$

$f_{N\setminus T}$ is normalized, monotone and submodular.
The polymatroid $P(f_{N\setminus T})$ is imbedded in the vector space $\mathbb{R}^{N\setminus T}$; moreover

$$P(f_{N\setminus T}) = P(f) \cap \{x \in \mathbb{R}^N \mid x_i = 0 \text{ for } i \in T\}.$$

2. We can treat and eliminate the following cases:
if $u \notin \mathbb{R}^N_+$, $\lambda_{max} = 0$.
If there exists $i \in N$ with $u_i = 0$, $P(f) \cap \Delta_u = P(f_{N\setminus\{i\}}) \cap \Delta_u$ and we can replace N by $N \setminus \{i\}$.
Finally, assume that there exists $i \in N$ with $f(\{i\}) = 0$; for any feasible point $x \in P(f)$, $x_i = 0$ since $0 \leq x_i \leq f(\{i\})$ and $[0, \lambda_{max}]u = P(f_{N\setminus\{i\}}) \cap \Delta_u$.
If $u_i > 0$, $\lambda_{max} = 0$.
If $u_i = 0$, we can replace N by $N \setminus \{i\}$.
So, we will assume from now on that $u > 0$ and that $f(\{i\}) > 0$ for all $i \in N$.

3. By order we mean in this paper a subset of N, called the support of the order, and a total order on this support. Elements in the support will be called ordered elements and elements not in the support will be called singletons.
$\{\mathcal{O}^0, \mathcal{O}^1, \ldots, \mathcal{O}^k, \ldots, \mathcal{O}^{\overline{k}}\}$ is the set of distinct orders, $\mathcal{K} = \{0, 1, \ldots, k, \ldots, \overline{k}\}$ is the set of upper indices listing the distinct orders.
The order relation of $\mathcal{O}^k$ will be denoted $\prec_k$. We will assume that, in the description of $\mathcal{O}^k$, the elements of the support S^k are sorted in increasing order with respect to the order relation $\prec_k$:

$$\mathcal{O}^k = [i_1, \ldots, i_l, \ldots, i_m].$$

It was proved by Edmonds [6] that extreme points of $P(f)$ can be generated by the greedy algorithm: given an order $\mathcal{O}^k = [i_1, \ldots, i_m]$ with support S^k, the point $a^k \in \mathbb{R}^N$ whose components are:

$$\begin{aligned} a^k_{i_1} &= f(\{i_1\}) \\ a^k_{i_l} &= f(\{i_1, \ldots, i_l\}) - f(\{i_1, \ldots, i_{l-1}\}) \text{ for } l = 2, \ldots, m \\ a^k_i &= 0 \text{ for all } i \in N \setminus S^k. \end{aligned}$$

is an extreme point of $P(f)$. a^k is the extreme point generated by $\mathcal{O}^k$. Edmonds proved also that any extreme point of $P(f)$ is generated by an order. Note also that

$$a^k(\{i_1, \ldots, i_l\}) = f(\{i_1, \ldots, i_l\}) \text{ for } l = 1, \ldots, m. \tag{1}$$

Two distinct orders may generate the same extreme point. In [1] Bixby, Cunningham, and Topkis gave a complete characterisation of extreme points of $P(f)$. In particular they proved that the intersection of all the orders which

generate the same extreme point is a minimal partial order and two minimal distinct partial orders generate two distinct extreme points. However we will not use this result here.

4. We will associate to a matroid order $\mathcal{O}^k$ the following oriented graph $G(k)$:
The node-set of $G(k)$ is $\{0\} \cup N$ where 0 is a new element; from now on we will use the term element rather than node.
The edge-set of $G(k)$ denoted $E(k)$ is defined as follows:

* $e = (0,i)_k \in E(k)$ if and only if i is a singleton of $\mathcal{O}^k$,
* $e = (i,j)_k \in E(k)$ if and only if i and j are ordered elements of $\mathcal{O}^k$ and $i \prec_k j$.

If $K \subseteq \mathcal{K}$ we define the graph $G(K)$ as:

$$G(K) = (\{0\} \cup N, \bigcup_{k \in K} E(k)).$$

Note that $G(K)$ may have (many) parallel edges:
if $i \prec_{k_1} j$ and $i \prec_{k_2} j$ for $k_1, k_2 \in K$, $(i,j)_{k_1}$ and $(i,j)_{k_2}$ are parallel edges.

5. We will now define two operations on the order $\mathcal{O}^k = [i_1, \ldots, i_m]$.
If j is a singleton of $\mathcal{O}^k$, $\mathcal{O}^{k'}$ is obtained by inserting j after the last element i_m of $\mathcal{O}^k$ and $\mathcal{O}^{k'}$ will be called the order associated to the edge $e = (0,j)_k$.
If $i \prec_k j$. $\mathcal{O}^{k'}$ is obtained by moving j just before i in $\mathcal{O}^k$and $\mathcal{O}^{k'}$ will be called the order associated to the edge $e = (i,j)_k$. In both cases, we say that $\mathcal{O}^{k'}$ is an adjacent order to $\mathcal{O}^k$.
Let us set $b = a^{k'} - a^k$. We recall (see for instance [14] or [20]) and prove for completeness the following result:

Lemma 1. *If $\mathcal{O}^{k'}$ is associated to $e = (0,j)_k$, $b_j \geq 0$; $b_i = 0$ for $i \in N \setminus \{j\}$.*
If $\mathcal{O}^{k'}$ is associated to $e = (i,j)_k$, $b_j \geq 0$; $b_l \leq 0$ for $i \prec_k l \prec_k j$ or $l = i$, $b_l = 0$ otherwise. Moreover $\sum_{i=1}^n b_i = 0$.

Proof: Set $T_s = \{r \,|r \in N,\ r \prec_k s\}$ for $s \in N$ and let S^k be the support of $\mathcal{O}^k$.
If $\mathcal{O}^{k'}$ is associated to $e = (0,j)_k$, $\quad b_i = 0 \quad$ for $i \in N \setminus \{j\}$.
As f is monotone, $b_j = {a_j}^{k'} - {a_j}^k = f(S^k \cup \{j\}) - f(S^k) \geq 0$.
If $\mathcal{O}^{k'}$ is associated to $e = (i,j)_k$, as $T_i \subset T_j$ and f is submodular,

$$b_j = {a_j}^{k'} - {a_j}^k = f(T_i \cup \{j\}) - f(T_i) - (f(T_j \cup \{j\}) - f(T_j)) \geq 0.$$

If $i \prec_k l \prec_k j$ or $l = i$,

$$b_l = {a_l}^{k'} - {a_l}^k = f(T_l \cup \{j,l\}) - f(T_l \cup \{j\}) - (f(T_l \cup \{l\}) - f(T_l)) \leq 0.$$

$\mathcal{O}^k$ and $\mathcal{O}^{k'}$ have the same support S^k and $\sum_{i=1}^n a_i^k = \sum_{i=1}^n a_i^{k'} = f(S^k)$.
Thus $\sum_{i=1}^n b_i = 0$.

6. We will express points of the half-line Δ_u as convex combinations of extreme points of $P(f)$. So, let $K \subseteq \mathcal{K}$. We say that K is a feasible set if the system:

$$\mathcal{L}(K) \qquad \begin{cases} \alpha_k \geq 0 \quad \text{for} \quad k \in K \\ \lambda u = \sum_{k \in K} \alpha_k a^k \\ \sum_{k \in K} \alpha_k = 1 \end{cases}$$

has a feasible solution. If K is feasible and the vectors $a^k : k \in K$ are linearly independent, $|K| \leq n$ and the solution $(\alpha_k,\ k \in K;\ \lambda)$ of $\mathcal{L}(K)$ is unique. If this solution is strictly positive, K is called a positive basis. λ will be called the value of the basis. If K is a feasible set, K contains a positive basis (this is Caratheodory's theorem).
A set $\{b^1, \ldots, b^n\}$ of vectors of $\mathbb{R}^n$ is called triangular if:

* $b_i^k = 0$ whenever $1 \leq k < i \leq n$,
* $b_i^k \leq 0$ whenever $1 \leq i < k \leq n$,
* $b_k^k > 0$ for $k = 1, \ldots, n$.

If $\{b^1, \ldots, b^n\}$ is triangular, the $n \times n$ matrix B induced by this set of column-vectors is upper triangular and non-singular since $b_k^k > 0$ for $k = 1, \ldots, n$. Thus the linear system $\mu_1 b_1 + \ldots + \mu_n b_n = u > 0$ has a unique solution $\mu = (\mu_1, \ldots, \mu_n)$ and by standard results in linear algebra the inverse of B is non-negative and $\mu > 0$.
This solution can be computed in $O(n^2)$ time.
Let K be a positive basis and let K_1 be a subset of K. Assume that there exists $K' \subseteq \mathcal{K}$ and a triangular set $\{b^1, \ldots, b^n\}$ with
$b^r = a^{k'_r} - a^{k_r}$ where $k'_r \in K'$ and $k_r \in K_1$ for $r = 1, \ldots, n$.
We can now establish the following easy but fundamental lemma:

Lemma 2. *There exists a positive basis contained in* $(K \cup K') \setminus \{k_1\}$ *for some* $k_1 \in K_1$.

Proof: As $\{b^1, \ldots, b^n\}$ is a triangulated set, there exists n non-negative numbers μ_r, $r = 1, \ldots, n$ such that

$$u = \sum_{r=1}^n \mu_r a^{k'_r} - \sum_{r=1}^n \mu_r a^{k_r}.$$

So, there exists non-negative numbers ν_k, $k \in K_1 \cup K'$ with :

$$\begin{cases} u = \sum_{k\in K'} \nu_k a^k - \sum_{k\in K_1} \nu_k a^k \\ \sum_{k\in K'} \nu_k - \sum_{k\in K_1} \nu_k = 0. \end{cases}$$

Let $(\alpha_k,\ k \in K; \lambda)$ be the solution of $\mathcal{L}(K)$. For any $\lambda' > 0$,

$$\begin{cases} (\lambda+\lambda')u = \sum_{k\in K} \alpha_k a^k + \lambda'(\sum_{k\in K'} \nu_k a^k - \sum_{k\in K_1} \nu_k a^k) \\ \sum_{k\in K} \alpha_k + \lambda'(\sum_{k\in K'} \nu_k - \sum_{k\in K_1} \nu_k) = 1 \end{cases}$$

Set $\lambda' = \min_{k\in K_1} \frac{\alpha_k}{\nu_k}$ (if $\nu_k = 0$, $\frac{\alpha_k}{\nu_k} = +\infty$) and let $k_1 \in K_1$ be an upper indice which realizes this minimum. The set $(K \cup K') \setminus \{k_1\}$ is feasible and contains a positive basis.
$\mathcal{L}((K \cup K') \setminus \{k_1\})$ has $n+1$ rows and at most $2n$ columns. By standard results in linear programming, we can find the new feasible basis in a time $O(n^3)$ (See for instance [14], chapter 4, exercise 5).

7. Our initial positive basis will be $K = \{1, \ldots, n\}$ where $\mathcal{O}^k = [k]$ for $i = 1, \ldots, n$. Thus $a^k_i = 0$ if $i \neq k$ and $a^k_k = f(\{k\})$;
the solution associated to this basis is:

$$\alpha_k := \frac{\lambda u_k}{f(\{k\})}\ k = 1, \ldots, n; \qquad \lambda := \frac{1}{\sum_{k=1}^n \frac{u_k}{f(\{k\})}}.$$

3 The Intersection Algorithm

Let $(\alpha_k \text{ for } k \in K;\ \lambda)$ be the solution associated to a positive basis K. $\lambda u(T) \leq f(T)$ for any $T \subseteq N$; equality holds for λ_{max} and $T_{min} \subseteq N$ with

$$\frac{f(T_{min})}{u(T_{min})} = \min\left(\frac{f(T)}{u(T)} \mid \emptyset \subset T \subseteq N\right).$$

Optimality occurs when the following condition holds:

Proposition 1. *If 0 is not a root of $G(K)$, let T be the set of elements of N not reachable from 0; $\lambda_{max} = \lambda$ and $T_{min} = T$.*

Proof: If 0 is not a root of $G(K)$, let T be the set of elements of N not reachable from 0. Let $\mathcal{O}^k = [i_1, \ldots, i_m]$ for some $k \in K$.
If $(0, i)_k$ is an edge of $G(K)$, i is a singleton of $\mathcal{O}^k$ and $i \notin T$;
so $T \subseteq \{i_1, \ldots, i_m\}$. If $r \prec_k l$ and $l \in T$, then $r \in T$. This implies that $T = \{i_1, i_2, \ldots, i_l\}$ for some $1 \leq l \leq m$; equations (1) imply that $a^k(T) = f(T)$.
$\lambda u(T) = \sum\limits_{k\in K} \alpha_k a^k(T) = (\sum\limits_{k\in K} \alpha_k) f(T) = f(T)$ and our lemma is proved.

So we will study now the case where 0 is a root of $G(K)$.
As the set K will be updated at each iteration of the algorithm, we can always assume that $K = \{1, 2, \ldots, |K|\}$.
Let $d_1(i)$ denote the distance from 0 to i for each $i \in N$. $d_1(i)$ will be the first parameter assigned to i.
We say that an element $j \in N$ is in bad position on $\mathcal{O}^k$ if, either j is a singleton of $\mathcal{O}^k$, or if there exists an element i with $i \prec_k j$ and $d_1(i) < d_1(j)$. The number of elements in bad position on $\mathcal{O}^k$ will be denoted $\alpha(k)$ and we set:
$K(j) = \{k \in K \mid j \text{ is in bad position on } \mathcal{O}^k\}$.
Our second parameter $d_2(j)$ is defined by: $d_2(j) = \max_{k \in K(j)} \alpha(k)$.
We say that $k \in K(j)$ is critical for j if $d_2(j) = \alpha(k)$ and we say that k is a critical upper-indice of K if k is critical for at least one element of N.
K_1 will denote the set of critical indices.
Finally the third parameter $d_3(j)$ assigned to j is:
$d_3(j) = k_j = \min(k \mid k \text{ is critical for } j)$.
We will also associate to K three parameters:
$\Delta_1(K) = \sum_{j=1}^{n} d_1(j)$; $\quad \Delta_2(K) = \sum_{j=1}^{n} d_2(j)$; $\quad \Delta_3(K) = |K_1|$.
Before describing our algorithm, we need to implement a special procedure which renames the elements of N according to the following rules:

* $d_1(i) < d_1(j) \Longrightarrow i < j$ (rule 1)
* $d_1(i) = d_1(j)$, $d_2(i) > d_2(j) \Longrightarrow i < j$ (rule 2)
* $d_1(i) = d_1(j)$, $d_2(i) = d_2(j)$, $d_3(i) < d_3(j) \Longrightarrow i < j$ (rule 3)
* $d_1(i) = d_1(j)$, $d_2(i) = d_2(j)$, $d_3(i) = d_3(j) = k$, $i \prec_k j \Longrightarrow i < j$ (rule 4)

Note that there exists a unique way to rename the elements of N if we apply these four rules (except for singletons which can be renamed in many ways since rule 4 do not apply to singletons).
Let $j \in N$; if $d_1(j) = 1$, j is a singleton for $\mathcal{O}^{k_j}$ and we will associate to j the edge $e_j = (0, j)_{k_j}$. If $d_1(j) > 1$, j is in bad position on $\mathcal{O}^{k_j}$ for some $k_j \in K$; there exists an element $i \in N$ with $i \prec_{k_j} j$ and $d_1(i) < d_1(j)$ and we can choose for i the smallest predecessor of j on $\mathcal{O}^{k_j}$ which satisfies this property; we will associate to j the edge $e_j = (i, j)_{k_j}$.
Let $\mathcal{O}^{k'_j}$ be the adjacent orders induced by e_j for $j \in N$ and consider the set of vectors $\{b^1, \ldots, b^n\}$ with $b^j = a^{k'_j} - a^{k_j}$ for $j = 1, \ldots, n$.
The following result holds:

Lemma 3. *Either one of the vectors $b^1, \ldots, b^n$ is the null vector, or the set $\{b^1, \ldots, b^n\}$ is triangulated.*

Proof: Assume that $b^j \neq 0$ for $j = 1, \ldots, n$.
If $e_j = (0, j)_{k_j}$, $b^j_j \geq 0$ and $b^j_i = 0$ for $i \neq j$ by lemma 1. As $b^j \neq 0$, $b^j_j > 0$.

If $e_j = (i,j)_{k_j}$, $b^j_j \geq 0$ but $\sum_{l=1}^n b^j_l = 0$ by lemma 1 and $b^j_l \leq 0$ for $l \neq j$ by lemma 1, therefore $b^j_j > 0$. As $d_1(i) < d_1(j)$, $i < j$ by rule 1.
Let $l \in N$ distinct from i and j.
If l does not satisfy the relation $i \prec_{k_j} l \prec_{k_j} j$, $b^j_l = 0$;
if $i \prec_{k_j} l \prec_{k_j} j$, $b^j_l \leq 0$ by lemma 1.
But $d_1(l) = d_1(j) - 1$ or $d_1(l) = d_1(j)$;
if $d_1(l) = d_1(j) - 1$, then $l < j$ (rule 1).
if $d_1(l) = d_1(j)$, l is in bad position on $\mathcal{O}^{k_j}$; $d_2(l) \geq \alpha(k_j) = d_2(j)$.
If $d_2(l) > d_2(j)$, then $l < j$ (rule 2);
if $d_2(l) = d_2(j)$, k_j is critical for l and $k_l \leq k_j$. If $k_l < k_j$, then $l < j$ (rule 3).
If $k_l = k_j$, $l \prec_{k_j} j$, then $l < j$ (rule 4).
So, for all possible situations, we have $b^j_l \leq 0$ for $1 \leq l < j$, $b^j_j > 0$ and $b^j_l = 0$ for $j < l \leq n$; this proves our statement.
We propose now a strongly polynomial algorithm to compute λ_{max}:

INTERSECTION ALGORITHM

Input: A normalized, monotone, submodular function f; a direction $u \in \mathbb{R}^N$.
Output: λ_{max}, T_{min}.

step 1. Take for initial positive basis the set $K := \{1, \ldots, n\}$ and compute the initial solution $(\alpha_1, \ldots, \alpha_n, \lambda)$ associated to this basis.
step 2. Represent $G(K)$ by the list of the set of out-neighbours of each node of $G(K)$.
step 3. If 0 is not a root, let T be the set of elements not reachable from 0, $\lambda_{max} = \lambda$, $T_{min} = T$;
return λ_{max}, T_{min}; **then stop**.
else compute the distance $d_1(j)$ from 0 to each element $j \in N$.
step 4. Compute for each $k \in K$ the number $\alpha(k)$ and for each element $j \in N$ the parameters $d_2(j)$ and $d_3(j) = k_j$.
Compute the numbers $\Delta_1(K)$, $\Delta_2(K)$, $\Delta_3(K)$.
step 5. Implement the relabeling procedure on the set N using rules 1,2,3,4.
step 6. Find the edges $e_j := (i,j)_{k_j}$ and the vectors $b^j := a^{k'_j} - a^{k_j}$ for $j \in N$.
step 7. If one of this vector say $b^j = a^{k'_j} - a^{k_j}$ is the null vector, set $K' := K \cup \{k'_j\}$, $K := K' \setminus \{k_j\}$ and **go to step 2**.
else set $K' := \{k'_1, \ldots, k'_n\}$ and K_1 be the set of critical indices of K; apply the procedure described in lemma 2. Let $K_2 \subseteq (K \cup K') \setminus \{k_j\}$ for some $k_j \in K_1$ be the positive basis obtained after application of this procedure; $K := K_2$, **go to step 2**.

It remains to analyze the complexity of this algorithm. First, we study the running time of each step of an iteration. The most critical step is **step 2** where we have to represent the graph $G(K)$ which may have $O(n^3)$ edges.
Let L be the $(n+1) \times (n+1)$ adjacency matrix of $G(K)$ where $L(i,j) = 1$ if j is an out-neighbour of i and $L(i,j) = 0$ otherwise. We start to build the adjacency matrix L_k for each graph $G(\{k\})$ $(k \in K)$: first we read the sequence $\mathcal{O}^k = [i_1, \ldots, i_m]$ in $O(n \log n)$ time. Then we fill the coefficients $L_k(0,j)$ for $1 \leq j \leq n$ in $O(n)$ time (these coefficients are associated to singletons). To get the rest of the adjacency matrix L_k we start from the $m \times m$ adjacency matrix associated to the ordered set: $[1, \ldots, m]$. This matrix is upper triangular with coefficients equal to 1 above the diagonal and 0 otherwise and can be obtained in $O(n^2)$ time. Permuting the rows and the columns of this matrix requires $O(n^2)$ elementary operations. The permutation which produces the matrix L_k is obtained by sorting the elements of $\mathcal{O}^k = [i_1, \ldots, i_m]$ by increasing value of the indices in time $O(n \log n)$. So the global time is $O(n^2)$. All the matrices L_k are obtained in $O(n^3)$ time since $|K| \leq n$.
Now, $L(i,j) = 1$ if, for some $k \in K$, $L_k(i,j) = 1$ and $L(i,j) = 0$ otherwise; each coefficient $L(i,j)$ is computed in $O(n)$ time. Thus the total running time for obtaining L is $O(n^3)$.
Using the adjacency matrix for $G(K)$, the running time of **step 3** is $O(n^2)$.
in **step 4** each $\alpha(k)$ is computed in $O(n^2)$ and all the $\alpha(k)$ in $O(n^3)$.
All parameters $d_2(j)$ and $d_3(j)$ for $j \in N$ can be computed in $O(n^2)$ time.
The three numbers Δ_1, Δ_2, Δ_3 are obtained in linear time. So, the total amount of work for **step 4** is $O(n^3)$.
The relabeling procedure of **step 5** can be implemented in $O(n^2)$ time: we first have to sort the elements of N by increasing distance, then by decreasing value of the second parameter $d_2(j)$, then by increasing value of the third parameter k_j and finally by describing the list of the ordered elements of $\mathcal{O}^{k_j}$.
In **step 6** we have n^2 oracle calls and a time $O(\gamma n^2)$ if γ is the time for an oracle call. We compute each vector $a^{k'_j}$ in $O(n^2)$ time. Thus, the total amount of work to implement all the parts of **step 6** is $O(n^3 + \gamma n^2)$.
We already noticed in lemma 2 that **step 7** can be implemented in $O(n^3)$ time.
As a consequence, the time needed to perform all the steps of the current iteration is $O(n^3 + \gamma n^2)$.
We need now to find the maximum number of iterations of the algorithm. Let K and $\overline{K}$ be the positive bases in two successive iterations of the algorithm. The analysis of the complexity of the algorithm is based on the three following claims:

Claim. $\Delta_1(\overline{K}) \geq \Delta_1(K)$

Proof: Let $e_j = (i,j)_{k_j}$ be the edge associated to $j \in N$ in **step 6**; $\mathcal{O}^{k'_j}$ is the adjacent order of $\mathcal{O}^{k_j}$. Let $(l,l')_{k'_j}$ be an edge of $G(k'_j)$.
If $(l,l')_{k_j}$ is also an edge of $G(k_j)$, $d_1(l') - d_1(l) \leq 1$;
if $(l,l')_{k_j}$ is not an edge of $G(k_j)$, there are two possible cases:
either j is a singleton of $\mathcal{O}^{k_j}$ and $l' = j$; as $d_1(j) = 1$, $d_1(l') - d_1(l) \leq 1$.
or j is an ordered element of $\mathcal{O}^{k_j}$, $l = j$ and $l' = i$ or $i \prec_{k_j} l' \prec_{k_j} j$; but $d_1(i) = d_1(j) - 1 \leq d_1(l') \leq d_1(j)$ and $d_1(l') - d_1(l) \leq 1$.
So, the relation $d_1(l') - d_1(l) \leq 1$ always hold. This shows that the distances in $G(K)$ and in $G(K \cup \{k'_j\})$ are equal. $G(K \cup K')$ is obtained by adding successively to $G(K)$ all the edge-sets of the graphs $G(k'_j)$ for all $k'_j \in K'$. Thus the distances are the same in $G(K \cup K')$ and in $G(K)$. But $G(\overline{K})$ is obtained from $G(K \cup K')$ by deletion of a subset of edges. Hence the distances can only increase in $G(\overline{K})$ and $\Delta_1(\overline{K}) \geq \Delta_1(K)$.

Claim. If $\Delta_1(\overline{K}) = \Delta_1(K)$, then $\Delta_2(\overline{K}) \leq \Delta_2(K)$

Proof: Let $e_j = (i,j)_{k_j}$ be the edge of the previous claim. As the distances are equal in $G(K)$ and $G(\overline{K})$ j is in bad position on $\mathcal{O}^{k_j}$ but is not in bad position on $\mathcal{O}^{k'_j}$; this is obvious if j is a singleton; this is also true if j is not a singleton by our choice of i; indeed recall that i is the first element such that $d_1(i) < d_1(j)$ and $i \prec_{k_j} j$ when we describe $\mathcal{O}^{k_j} = [i_1, \ldots, i_m]$ from i_1 to i_m. Elements distinct from j are in bad position on $\mathcal{O}^{k'_j}$ if and only if they are in bad position on $\mathcal{O}^{k_j}$. Thus $\alpha(k'_j) = \alpha(k_j) - 1$.
For an element l in bad position on $\mathcal{O}^{k_j}$ and $\mathcal{O}^{k'_j}$, we have $d_2(l) \geq \alpha(k_j) > \alpha(k'_j)$. No upper indice $k \in K'$ can be critical in the set $K \cup K'$; the parameters $d_2(l)$ remain unchanged when we replace $G(K)$ by $G(K \cup K')$. But $G(\overline{K})$ is obtained from $G(K \cup K')$ by deletion of a subset of upper indices. Therefore, either $d_2(l)$ remains unchanged or $d_2(l)$ strictly decreases. This proves the claim.

Claim. If $\Delta_1(\overline{K}) = \Delta_1(K)$ and $\Delta_2(\overline{K}) = \Delta_2(K)$, then $\Delta_3(\overline{K}) < \Delta_3(K)$

Proof: Let us continue the proof of the preceding claim. If the parameters $d_2(j)$ are unchanged and l is in bad position on $\mathcal{O}^{k'_j}$, $d_2(l) \geq \alpha(k_j) > \alpha(k'_j)$. Hence k'_j is not critical for l and k'_j cannot be a critical indice of $\overline{K}$. The set of critical indices of $\overline{K}$ is included in K_1; but there exists at least a critical indice k_j of K_1 which does not belong to $\overline{K}$; this implies that $\Delta_3(\overline{K}) < \Delta_3(K)$.
We can state now our main result:

Theorem 1. *The running time of the Intersection Algorithm is $O(n^8 + \gamma n^7)$, where γ is the time for an oracle call.*

Proof: Define a lexicographic order among the positive bases enumerated during the algorithm. We say that $K \prec \overline{K}$ if $\Delta_1(\overline{K}) > \Delta_1(K)$ or

$\Delta_1(\overline{K}) = \Delta_1(K)$, $\Delta_2(\overline{K}) < \Delta_2(K)$ or $\Delta_1(\overline{K}) = \Delta_1(K)$, $\Delta_2(\overline{K}) = \Delta_2(K)$, $\Delta_3(\overline{K}) < \Delta_3(K)$. By the preceding claims these bases are totally ordered with the lexicographic order; hence the triplets $(\Delta_1(K), \Delta_2(K), \Delta_3(K))$ are all different.
But the value of a distance cannot exceed n and $\Delta_1(K) \leq n^2$; the value of each second parameter cannot exceed n and $\Delta_2(K) \leq n^2$; finally $\Delta_3(K) \leq n$. so the total number of iterations cannot exceed n^5. The total running time is $O(n^8 + \gamma n^7)$.
Note that this is precisely the running time of the original algorithms for submodular function minimization. It may be possible to improve this complexity bound by, for instance, updating informations from an iteration to the next iteration. It may also be possible that the number of iterations is not greater than n^4 but we did not investigate these questions.

4 The Parametric Intersection Algorithm

4.1 A parametric linear program

We consider in this section the following parametric linear program:

$$z(\lambda) = \max \left\{ \sum_{i \in N} x_i \mid x \in \mathrm{P(f)},\ x \leq \lambda u \right\} \qquad (\mathcal{P}_\lambda)$$

H_λ will be the polytope: $\{x \in \mathbb{R}^N; 0 \leq x \leq \lambda u\}$.
A subset $T \subseteq N$ is such that $z(\lambda) = \lambda u(N \setminus T) + f(T)$ for some $\lambda \geq 0$ will be called λ-tight.
The following two propositions are standard results. We give short proofs for completeness. For a generalization of Proposition 2, see theorem 7.15 of [10]. The property given in Proposition 3 corresponds to theorem 4.6 in Megiddo [15] and is also used in Fujishige's algorithm [9, 10] for finding the lexicographically optimal base of a polymatroid with respect to a weight vector.

Proposition 2. *There exists $\lambda_0 = 0 < \lambda_1 < \ldots < \lambda_s < \lambda_{s+1} = +\infty$ with $s \leq n$ and a unique increasing family of sets $T_0 = \emptyset \subset T_1, \ldots, \subset T_s = N$ such that:*

$$z(\lambda) = \lambda u(N \setminus T_r) + f(T_r) \text{ for } \lambda_r \leq \lambda \leq \lambda_{r+1},\ r = 0, \ldots, s.$$

Moreover T_r is the unique λ-tight set for $\lambda_r < \lambda < \lambda_{r+1},\ r = 0, \ldots, s$.

Proof: We will use the following theorem of Edmonds [6] (this theorem gives the rank formula of a polymatroid):

$$z(\lambda) = \min(f(T) + \lambda u(N \setminus T) |\, T \subseteq N).$$

The fonction $\lambda \in \mathbb{R}^+ \longrightarrow z(\lambda)$ which is the minimum of a finite set of non-decreasing affine functions is a piecewise-linear non-decreasing concave function. There exists $\lambda_0 = 0 < \lambda_1 < \ldots < \lambda_s$ and a family of sets $T_0, T_1, \ldots, T_s$ such that:

$$z(\lambda) = \lambda u(N \setminus T_r) + f(T_r) \text{ for } \lambda_r \leq \lambda \leq \lambda_{r+1}, \quad r = 0, \ldots, s-1.$$

For $0 \leq \lambda \leq \lambda_{max}$, $P(f) \cap H_\lambda = H_\lambda$; $x = \lambda u$ is the solution of $(\mathcal{P}_\lambda)$ and $z(\lambda) = \lambda u(N)$. For $\lambda > \lambda_{max}$, $\lambda u \notin P(f)$ and $\lambda u(N) > z(\lambda)$. This implies that $\lambda_0 = 0$, $\lambda_1 = \lambda_{max}$, $T_0 = \emptyset$.
For λ large enough, $H_\lambda \cap P(f) = P(f)$ and $z(\lambda) = f(N)$; so $z(\lambda) = f(N)$ for $\lambda \geq \lambda_s$ and $T_s = N$.
For $\lambda_1 \leq \lambda \leq \lambda_s$, let A and B be two λ-tight sets. By the submodularity of f, $A \cap B$ and $A \cup B$ are also λ-tight sets; hence we can assume that A (resp. B) is the unique λ-tight set of minimum (resp. maximum) cardinality and $A \subseteq B$.
If $A \subset B$, $u(N \setminus B) < u(N \setminus A)$ since $u > 0$; by the concavity and the continuity of $z(\lambda)$, there exists $\varepsilon > 0$ such that $z(\lambda') = f(A) + \lambda' u(N \setminus A)$ if $\lambda - \varepsilon \leq \lambda' \leq \lambda$ and $z(\lambda') = f(B) + \lambda' u(N \setminus B)$ if $\lambda \leq \lambda' \leq \lambda + \varepsilon$. Hence $z(\lambda)$ is not a linear function in any neighborhood of λ; thus $\lambda \in \{\lambda_1, \ldots, \lambda_s\}$.
If $\lambda = \lambda_r$ for $1 \leq r \leq s$, $T_{r-1} = A \subset B = T_r$.
Finally $r \leq n$ since the number of sets T_r cannot exceed n. This finishes the proof.
Let x^r be the solution of $(\mathcal{P}_{\lambda_r})$ and define a new direction:

$$u^r \begin{cases} u^r_i = 0 \text{ if } i \in T_r \\ u^r_i = u_i \text{ if } i \in N \setminus T_r \end{cases}$$

Consider the following half-line:

$$\Delta_{u^r} = \{x \in \mathbb{R}^N \mid x = x^r + (\lambda - \lambda_r)u^r,\ \lambda \geq \lambda_r\}.$$

We want to find now the extremities of the closed interval $P(f) \cap \Delta_{u^r}$.

Proposition 3. $\lambda_{r+1} = \max\{\lambda \mid x \in P(f),\ x = x^r + (\lambda - \lambda_r)u^r,\ \lambda \geq \lambda_r\}$.

Proof: If $x = x^r + (\lambda - \lambda_r)u^r$, $x(N) = x^r(N) + (\lambda - \lambda_r)u(N \setminus T)$.
But $x^r(N) = z(\lambda_r) = f(T_r) + \lambda_r u(N \setminus T_r)$.
Therefore $x(N) = f(T_r) + \lambda u(N \setminus T_r)$.
If x is the solution of the linear program defined in the statement of proposition 3, there exists a tight set $S \subseteq N$ for x which contains at least one element $i \in S \setminus T_r$. But $S \cup T_r$ which strictly contains T_r is also tight for x by the submodularity of f and $x(N) = f(S \cup T_r) + \lambda u(N \setminus (S \cup T_r))$.
As $x \in H_\lambda$, x is solution of $\mathcal{P}_\lambda$. But we know by proposition 2 that T_r is the largest λ_r-tight set and that T_r is the unique λ-tight set in the interval $\lambda_r < \lambda < \lambda_{r+1}$. Thus the set $S \cup T_r$ can exist only if $\lambda = \lambda_{r+1}$.

4.2 The Parametric Intersection Algorithm

Some of the ideas of our Parametric Intersection Algorithm appear in Fujishige [10]. We will study the case when the Intersection Algorithm returns $\lambda_1 = \lambda_{max}$ and the set T_{min}. The case λ_r for $r > 1$ is similar and will not be treated. $x^1 = \lambda_1 u$ is the solution of $\mathcal{P}_{\lambda_1}$; $(\alpha_k^1,\ k \in K;\ \lambda_1)$ is the solution of $\mathcal{L}(K)$ where K is the last positive basis K returned by the Intersection Algorithm:

$$\mathcal{L}(K) \qquad \begin{cases} \alpha_k \geq 0 \quad \text{for} \quad k \in K \\ \lambda u = \sum_{k \in K} \alpha_k a^k \\ \sum_{k \in K} \alpha_k = 1 \end{cases}$$

and T_1 is the set of nodes of $G(K)$ not reachable from 0. Instead of stopping as in the Intersection Algorithm, we could try to use these informations in order to find the second value λ_2. By proposition 3, λ_2 is the value of the linear program:

$$\lambda_2 = \max\{\lambda \mid x \in P(f),\ x = \lambda_1 u + (\lambda - \lambda_1)u^1,\ \lambda \geq \lambda_1\}$$

where u^1 is the new direction defined as in proposition 3. So our original system $\mathcal{L}(K)$ is changed into a different system:

$$\mathcal{L}^1(K) \qquad \begin{cases} \alpha_k \geq 0 \quad \text{for} \quad k \in K \\ \lambda_1 u + (\lambda - \lambda_1)u^1 = \sum_{k \in K} \alpha_k a^k \\ \sum_{k \in K} \alpha_k = 1 \end{cases}$$

Since λ_1 is fixed, the variables of $\mathcal{L}^1(K)$ are $(\alpha_k,\ k \in K;\ \lambda)$ and $(\alpha_k^1,\ k \in K;\ \lambda = 0)$ is the unique solution of $\mathcal{L}^1(K)$.
T_1 is the set of elements of $G(K)$ not reachable from 0 and $N_1 = N \setminus T_1$ is the set of elements reachable from 0; we will set $|T_1| = l \geq 1$. If $T_1 = N$, $\lambda_1 = \lambda_s$ and we stop. So, we will assume that $l < n$. $d_1(i)$ is the distance from 0 to i for $i \in N_1$. We observed in the proof of Proposition 1 that the elements of T_1 are the first elements of the sequence describing $\mathcal{O}^k$ for each $k \in K$: if

$$\mathcal{O}^k = [i_1, \ldots, i_l, i_{l+1}, \ldots, i_m]$$

$T_1 = \{i_1, \ldots, i_l\}$. To maintain this property throughout the next iterations of the algorithm, we assign to each element i of T_1 a first parameter $d_1(i)$ equal to n: if $j \in N$, $d_1(i) \geq d_1(j)$ and i will never be in bad position for any order $\mathcal{O}^k$ in the forthcoming iterations . The numbers $\alpha(k)$ for $k \in K$ are computed as in the Intersection Algorithm. Note however that no upper indice k is critical if $i \in T_1$, so we will set $d_2(i) = 0$ for $i \in T_1$. The third parameter will play no role for elements in T_1 and will be discarded.
Let $T_1 = \{n - l + 1, \ldots, n\}$. The elements of T_1 will never be affected by the

relabeling procedure because of the first rule of this procedure. Next, we will find the edges $e_j := (i,j)_{k_j}$ and the vectors $b^j := a^{k'_j} - a^{k_j}$ as in step 6. However these edges exist only for $j \in N_1$ since no indice k is critical for $j \in T_1$. Thus there exists $|N_1|$ edges e_j and $|N_1|$ vectors b^j. The elements of T_1 remain in the first position in any new order $\mathcal{O}^{k'_j}$ introduced in step 6. Since $b^j_i = 0$ for $i \in T_1$ and $j = 1, \ldots, n-l$ the linear system $\mu_1 b_1 + \ldots + \mu_n b_{n-l} = u^1$ has a non-negative solution. We can terminate step 7 as in the Intersection Algorithm.

PARAMETRIC INTERSECTION ALGORITHM

Input: A normalized, monotone, submodular function f; a direction $u \in \mathbb{R}^N$.
Output: $s \geq 1$; $\lambda_0, \ldots, \lambda_s$; $T_0, \ldots, T_s$.

step 1. Take for initial positive basis $K := \{1, \ldots, n\}$ and compute the initial solution $(\alpha_1, \ldots, \alpha_n, \lambda)$ associated to this basis. Set $r = 0$, $\lambda_0 = 0$, $T_0 = \emptyset$.
step 2. Represent $G(K)$ by the list of the set of out-neighbours of each node of $G(K)$.
step 3. If 0 is not a root, let T be the set of elements not reachable from 0; compute the distance $d_1(j)$ from 0 to each element $j \in N \setminus T$.
Set $d_1(j) = n$ and $d_2(j) = 0$ for $j \in T$.
if $T_r = T$ go to **step 4**
else $T_r \subset T$; set $r := r + 1$, $T_r := T$, $\lambda_r := \lambda$,
update $u \in \mathbb{R}^N$: $u_i := 0$ if $i \in T$. If $T = N$ **stop**.
else go to step 4.
step 4. Compute for each $k \in K$ $\alpha(k)$ and for each element $j \in N \setminus T$ the parameters $d_2(j)$ and k_j. Compute the numbers $\Delta_1(K)$, $\Delta_2(K)$, $\Delta_3(K)$.
step 5. Implement the relabeling procedure on the set $N \setminus T$ with rules 1,2,3,4.
step 6. Find the edges $e_j := (i,j)_{k_j}$ and the vectors $b^j := a^{k'_j} - a^{k_j}$ for $j \in N \setminus T$.
step 7. If one of this vector say $b^j = a^{k'_j} - a^{k_j}$ is the null vector, set $K' := K \cup \{k'_j\}$, $K := K' \setminus \{k_j\}$ and **go to step 2.**
else set $K' := \{k'_1, \ldots, k'_n\}$ and let K_1 be the set of critical indices of K; apply the procedure described in lemma 2. Let $K_2 \subseteq (K \cup K') \setminus \{k_j\}$ for some $k_j \in K_1$ be the positive basis obtained after application of this procedure; $K := K_2$ and **go to step 2**.

At the first iteration of the algorithm, all the elements of N are singletons and $K_1 = K$. Thus, the triplet associated to the initial basis is $(n, (n-1) \times$

$n, n)$. At the final iteration no element is reachable from 0 and the triplet is $(n^2, 0, 0)$. As in the proof of the Intersection Algorithm, all the bases are distinct and the number of iterations cannot exceed n^5; the complexity of the Parametric Intersection Algorithm is $O(n^8 + \gamma n^7)$. Let us give now two applications of this algorithm.

4.3 Strength of a polymatroid

Consider the last discontinuity λ_s of $z(\lambda)$:
$z(\lambda) = f(N) = f(N \setminus T_{s-1}) + \lambda_s u(T_{s-1})$.
$\lambda_s = \frac{f(N)-f(N\setminus T_{s-1})}{u(T_{s-1})}$.
If $(T, N \setminus T)$ is a partition of N with $f(N) > f(N \setminus T)$,
$f(N) = f(N \setminus T_{s-1}) + \lambda_s u(T_{s-1}) \leq f(N \setminus T) + \lambda_s u(T)$ and it is easy to see that:
$\lambda_s = \frac{f(N)-f(N\setminus T_{s-1})}{u(T_{s-1})} \geq \frac{f(N)-f(N\setminus T)}{u(T)}$.
If we set
$\sigma(f, u) = \min\{\frac{u(T)}{f(N)-f(N\setminus T)}$ for all $T \subset N$; such that $f(N) > f(N \setminus T)\}$,
$\sigma(f, u) = \frac{1}{\lambda_s}$.
By analogy with graphs, we call $\sigma(f, u)$ the strength of the polymatroid $P(f)$. If f is the rank function of a graphic matroid, $\sigma(f, u)$ is precisely the strength of a network.

4.4 Minimization of a submodular function

Let $\overline{f}$ be a submodular set function defined on N. $\overline{f}$ may not be monotone or normalized. The submodular function minimization problem consists in finding a subset of N which realizes the minimum of f. We will prove that this subset can be returned by the Parametric Intersection Algorithm for a well chosen value of the parameter λ.
Let us first make $(n + 2)$ oracle calls: $\overline{f}(i)$ for $i \in N$, $\overline{f}(N)$ and $\overline{f}(\emptyset)$.
Set $\delta = \max_{i\in N} \overline{f}(i)$ and $\delta_0 = \min(0, \overline{f}(N))$.

Lemma 4. *Let $\lambda = 2(n\delta - \delta_0) + 1$. Define f by $f(T) = \overline{f}(T) + \lambda|T| - \overline{f}(\emptyset)$, f is normalized, monotone and submodular.*

Proof: Clearly f is normalized and submodular.
Let $T \subseteq N$; By the submodularity of $\overline{f}$, $\overline{f}(T) \leq \sum_{i\in N} \overline{f}(\{i\}) \leq n\delta \leq n\delta - \delta_0$.
As $\delta_0 \leq \overline{f}(N) \leq n\delta$, $\lambda > 0$.
$\overline{f}(T) \geq \overline{f}(N) - \overline{f}(N \setminus T) \geq \delta_0 - n\delta$.
If S and T are two subsets of N, $-\lambda \leq \overline{f}(S) - \overline{f}(T) \leq \lambda$.
Assume that $S \subset T$:

$f(T) - f(S) \geq \overline{f}(T) - \overline{f}(S) + \lambda \geq 0$ and f is monotone.
Let u be the vector with all components equal to 1. If $S \subseteq N$ is λ-tight,

$$z(\lambda) = f(S) + \lambda|N \setminus S| = \min(f(T) + \lambda|N \setminus T| \text{ for all } T \subseteq N),$$

$\overline{f}(S)+\lambda|S|-\overline{f}(\emptyset)+\lambda|N\setminus S| = \min(\overline{f}(T)+\lambda|T|-\overline{f}(\emptyset)+\lambda|N\setminus T| \text{ for all } T \subseteq N)$.
This proves that $\overline{f}(S) = \min(\overline{f}(T)|\, T \subseteq N)$.
We just proved the following result:

Theorem 2. *The Parametric Intersection Algorithm solves the submodular function minimization problem and simultaneously computes the strength of the polymatroid in $O(n^8 + \gamma n^7)$ time.*

5 Conclusion

We were motivated in the study of the intersection of a polymatroid and a half-line by problems in the Telecommunication industry, suggested by Jérôme Galtier and Alexandre Laugier, where the strength of a network is a useful parameter; by choosing randomly the direction u according to a uniform distribution law it was possible to show that the average number of iterations for the Intersection Algorithm is $O(n)$; in a forthcoming paper, we will prove that the theoretical average complexity of a very simple algorithm for solving the Intersection Problem is $O(n^5)$, confirming the good practical bound observed by numerical tests.
The complexity of the Intersection Problem of a polytope with a straight line is trivial if the inequalities describing the polytope are explicitly given as input, by enumeration of these inequalities; if the polytope is given by its extreme points, the existence of a combinatorial algorithm to solve the Intersection Problem is a difficult open problem since it is possible to show that the existence of such an algorithm would answer positively the following question: does there exist a combinatorial, "simplex-like" algorithm for linear programming?
Finally the remaining case is when the polytope and the straight line are embedded in a vector space $\mathbb{R}^n$ with the number of inequalities describing the polytope not polynomial in n. The question is to find a strongly polynomial algorithm in n for solving the Intersection Problem. Of course a condition should be satisfied: the optimization problem on this polytope has to be solvable in polynomial time thanks to the ellipsoid method for instance. So we may hope that there exists strongly polynomial combinatorial algorithms when we replace the polymatroid by the matching polytope, the branching polytope, or the stable set polytope of a perfect graph. All these questions seem to be opened.

References

1. R.E. Bixby, W.H. Cunningham, and D.M. Topkis. The partial order of a polymatroid extreme point. *Mathematics of Operations Research*, 10(3):367-378, 1985.
2. E. Cheng and W.H. Cunningham. A faster algorithm for computing the strength of a network. *Information Processing Letters*, 49:209-212, 1994.
3. W.H. Cunningham. Testing membership in matroid polyhedra. *Journal of Combinatorial Theory*, Series B, 36:161-188, 1984.
4. W.H. Cunningham. Optimal attack and reinforcement of a network. *Journal of ACM*, 32(3): 549–561, 1985.
5. W.H. Cunningham. On submodular function minimization. *Combinatorica*, 5:185-192, 1985.
6. J. Edmonds. Submodular functions, matroids and certain polyhedra. *Combinatorial structures and their applications, Proceedings of the Calgary international Conference*, 69-87, Gordon and Breach, 1970.
7. L.K. Fleischer. Recent progress in submodular function minimization. *Optima*, 1-11, 2000.
8. L. Fleischer and S. Iwata. Improved algorithms for submodular function minimization and submodular flow. *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 107-116, 2000.
9. S. Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, 5: 186-196, 1980.
10. S. Fujishige. Submodular functions and optimization. *Annals of discrete mathematics*, 58. Elsevier, 2005.
11. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169-197, 1981.
12. S. Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM Journal on Computing*, 32:833-840, 2003.
13. S. Iwata, L. Fleisher, and S. Fujishige. A combinatorial strongly polynomial time algorithm for minimizing submodular functions. *Journal of the Association for Computing Machinery*, 48:761-777, 2001.
14. B. Korte and J. Vygen. Combinatorial Optimization: Theory and Algorithms. *Algorithms and Combinatorics 21*, Springer, 2008.
15. N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7: 97-107, 1974.
16. K. Nagano. A strongly polynomial algorithm for line search in submodular polyhedra. *Mathematical engineering technical report*, Department of Mathematical Informatics, The University of Tokyo, 2004.
17. K. Nagano. A faster parametric submodular function minimization algorithm and applications. *Mathematical engineering technical report*, Department of Mathematical Informatics, The University of Tokyo, 2007.
18. J.B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *IPCO*, 240-251, 2007.
19. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory*, Series B, 80(2):346-355, 2000.

20. A. Schrijver. Combinatorial optimization: polyhedra and efficiency, Springer, 2003.
21. A. Skoda. Force d'un graphe, multicoupes et fonctions sous-modulaires: aspects structurels et algorithmiques. PhD thesis, Université Pierre et Marie Curie, Paris 6, 2007.
22. J. Vygen. A note on Schrijver's submodular function minimization algorithm. *Journal of Combinatorial Theory B*, 88(2):399-402, 2003.